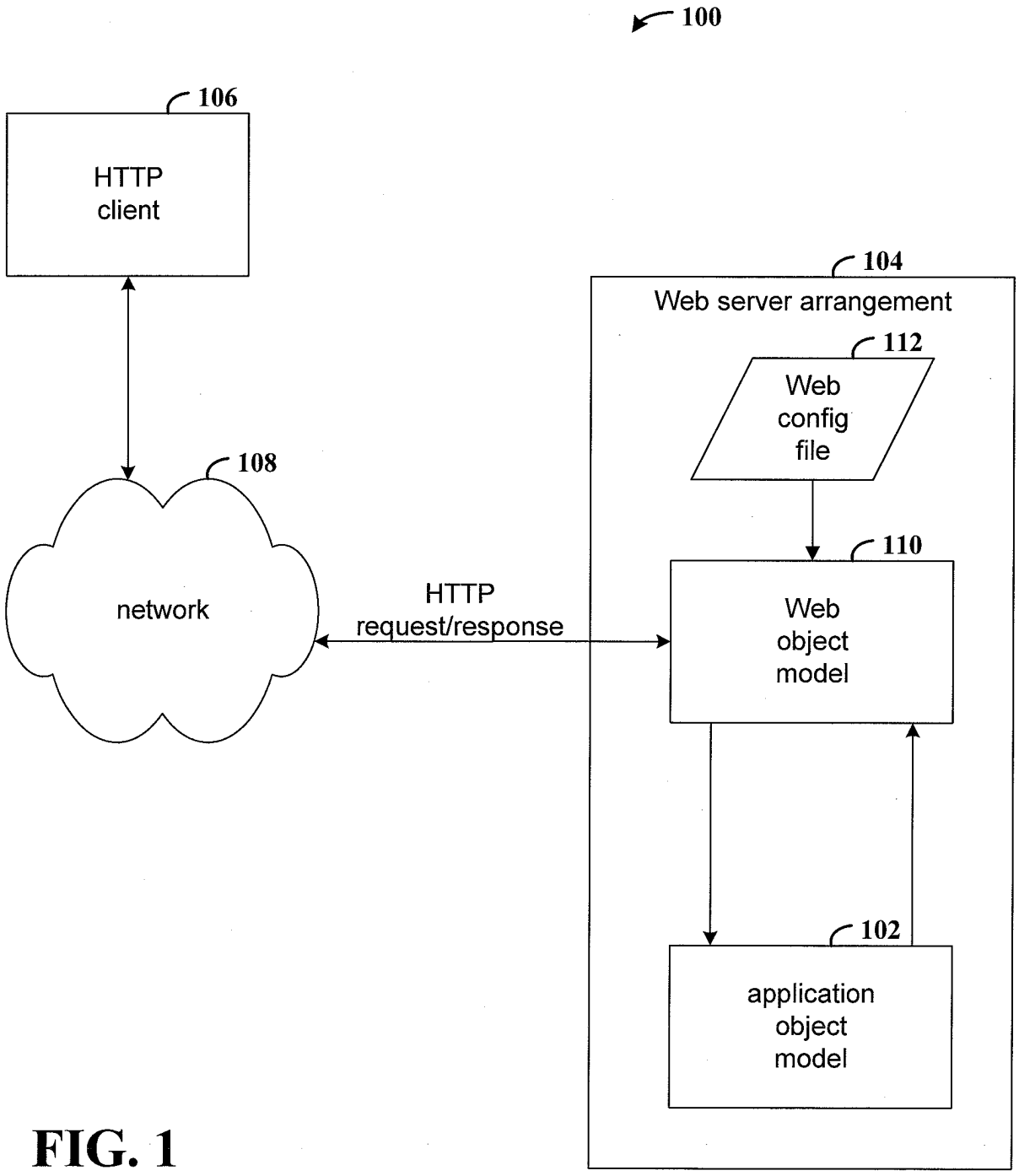
100
106
HTTP client
104
Web server arrangement
112
Web config file
108
network
HTTP request/response
110
Web object model
102
application object model

FIG. 1

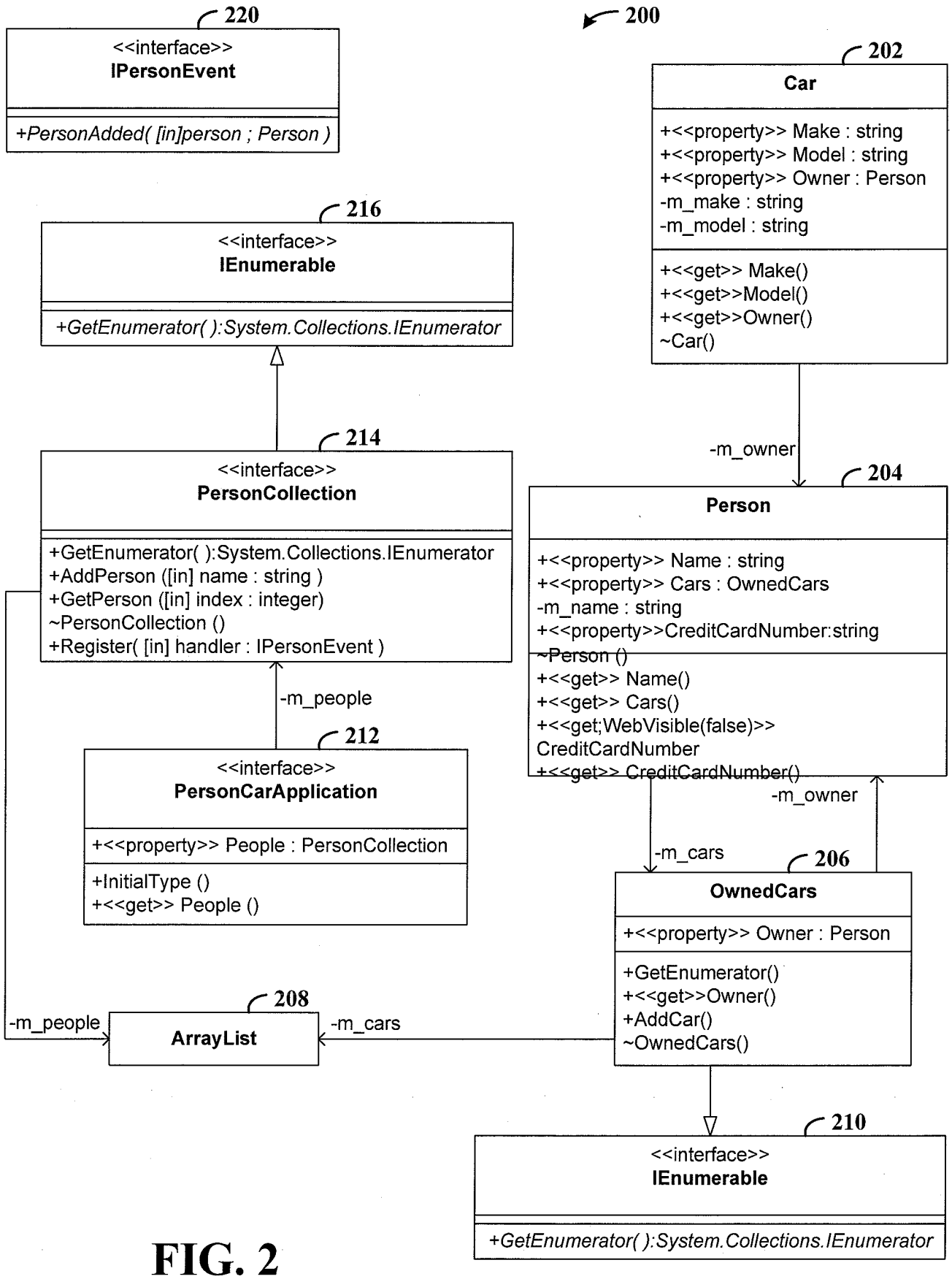
220
<<interface>>
IPersonEvent
+PersonAdded([in]person ; Person)
200
202
Car
+<<property>> Make : string
+<<property>> Model : string
+<<property>> Owner : Person
-m_make : string
-m_model : string
+<<get>> Make()
+<<get>>Model()
+<<get>>Owner()
~Car()
216
<<interface>>
IEnumerable
+GetEnumerator( ):System.Collections.IEnumerator
214
<<interface>>
PersonCollection
+GetEnumerator( ):System.Collections.IEnumerator
+AddPerson ([in] name : string)
+GetPerson ([in] index : integer)
~PersonCollection ()
+Register([in] handler : IPersonEvent)
-m_owner
204
Person
+<<property>> Name : string
+<<property>> Cars : OwnedCars
-m_name : string
+<<property>>CreditCardNumber:string
~Person ()
+<<get>> Name()
+<<get>> Cars()
+<<get;WebVisible(false)>>
CreditCardNumber
+<<get>> CreditCardNumber()
-m_people
212
<<interface>>
PersonCarApplication
+<<property>> People : PersonCollection
+InitialType ()
+<<get>> People ()
-m_owner
-m_cars
206
OwnedCars
+<<property>> Owner : Person
+GetEnumerator()
+<<get>>Owner()
+AddCar()
~OwnedCars()
208
-m_people
ArrayList
-m_cars
210
<<interface>>
IEnumerable
+GetEnumerator( ):System.Collections.IEnumerator

FIG. 2

300

```
Property Parsing
302 currentObject = initialObject; // obtained from Web Config File
304 currentPropertyName = GetNextPropertyNameFromURL();
306 while currentPropertyName != null
308     currentType = currentObject.GetType();
310     // See if current type contains a property with the current property name.
        if currentType.GetProperties() contains currentPropertyName then
312         currentObject = currentObject.GetProperty( currentPropertyName );
314     else
316         //No property name  with the current property name exists.  Assume
            //that property name is actually the value of an indexer
            indexerValue = currentPropertyName;
318         currentObject = currentObject.GetProperty( "Indexer", indexerValue );
320     currentPropertyName = GetNextPropertyNameFromURL();
322 methodDescription = ParseMethodDescriptionFromURL();
324 if methodDescription != null
326     currentType = currentObject.GetType();
328     method = currentObject.GetMethod (methodDescription);
330     currentObject = method.Invoke (currentObject, methodDescription);
332 if currentObject == null
334     returnValue = "";
336 else
338     currentType = currentObject.GetType();
340     if (IsValueType (currentTYpe)) then
342         returnValue = FormatValueType (currentObject);
344     else
346         returnValue = FormatReferenceType (currentObject);
348 response.Stream (returnValue);
```

FIG. 3

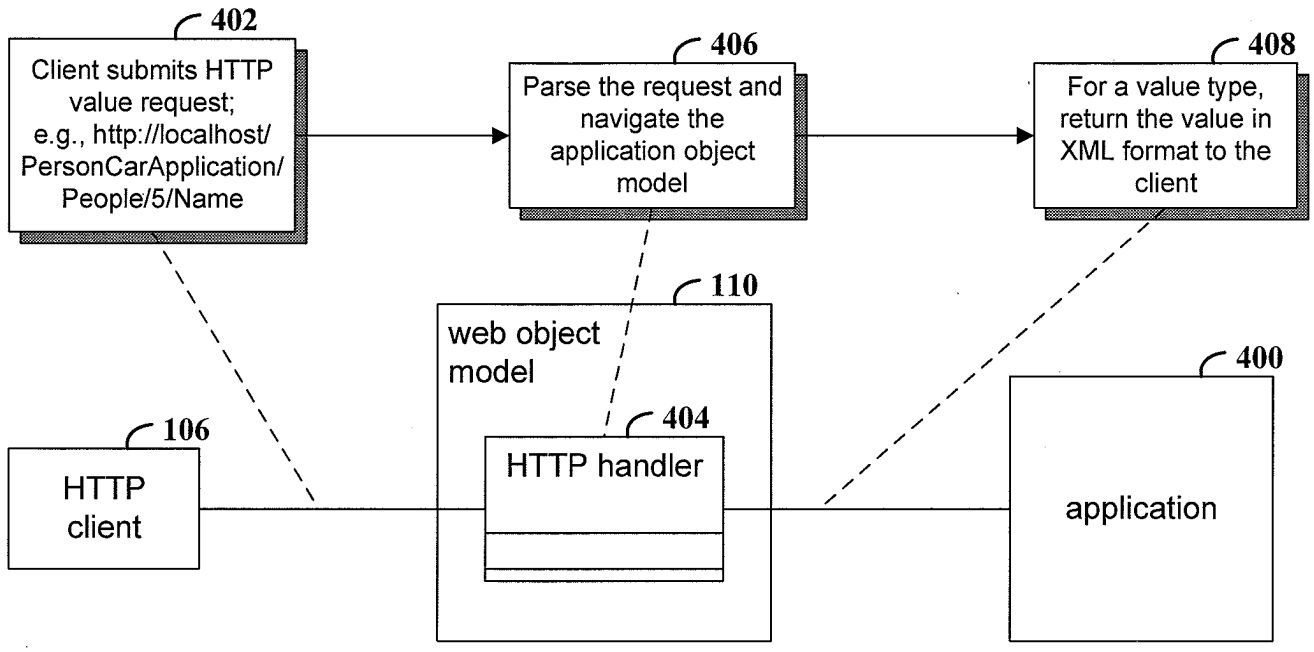
402
Client submits HTTP value request; e.g., http://localhost/ PersonCarApplication/ People/5/Name
406
Parse the request and navigate the application object model
408
For a value type, return the value in XML format to the client
110
web object model
404
HTTP handler
400
application
106
HTTP client

FIG. 4A

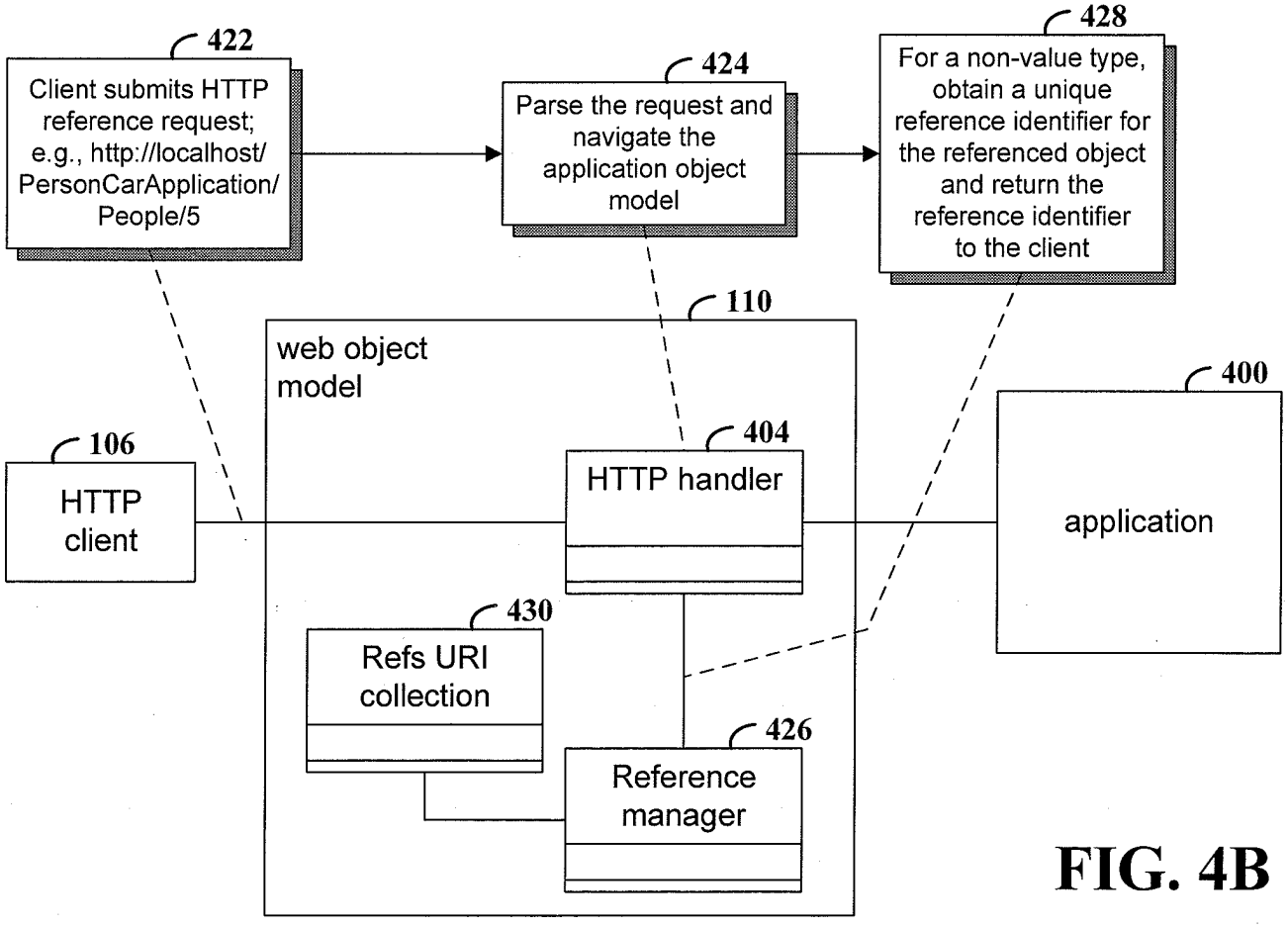
422
Client submits HTTP reference request; e.g., http://localhost/ PersonCarApplication/ People/5
424
Parse the request and navigate the application object model
428
For a non-value type, obtain a unique reference identifier for the referenced object and return the reference identifier to the client
110
web object model
400
application
106
HTTP client
404
HTTP handler
430
Refs URI collection
426
Reference manager

FIG. 4B

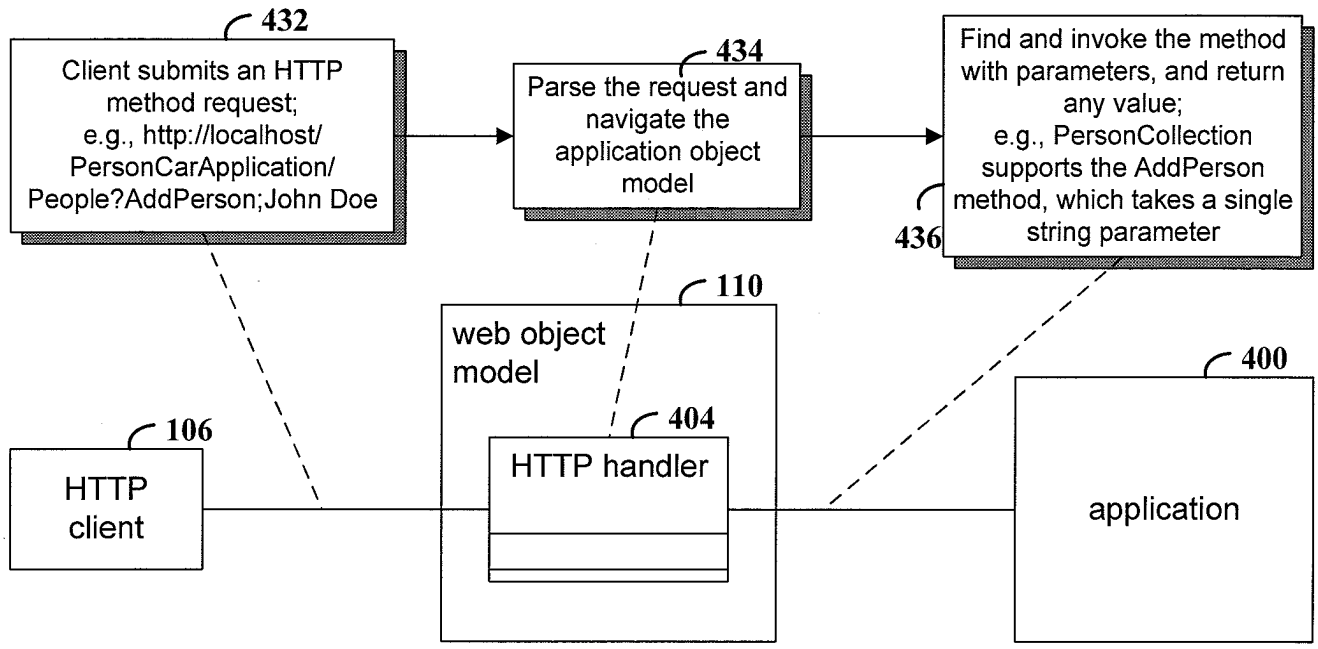
432
Client submits an HTTP method request; e.g., http://localhost/ PersonCarApplication/ People?AddPerson;John Doe
434
Parse the request and navigate the application object model
Find and invoke the method with parameters, and return any value; e.g., PersonCollection supports the AddPerson method, which takes a single string parameter
436
110
web object model
400
106
HTTP client
404
HTTP handler
application

FIG. 4C

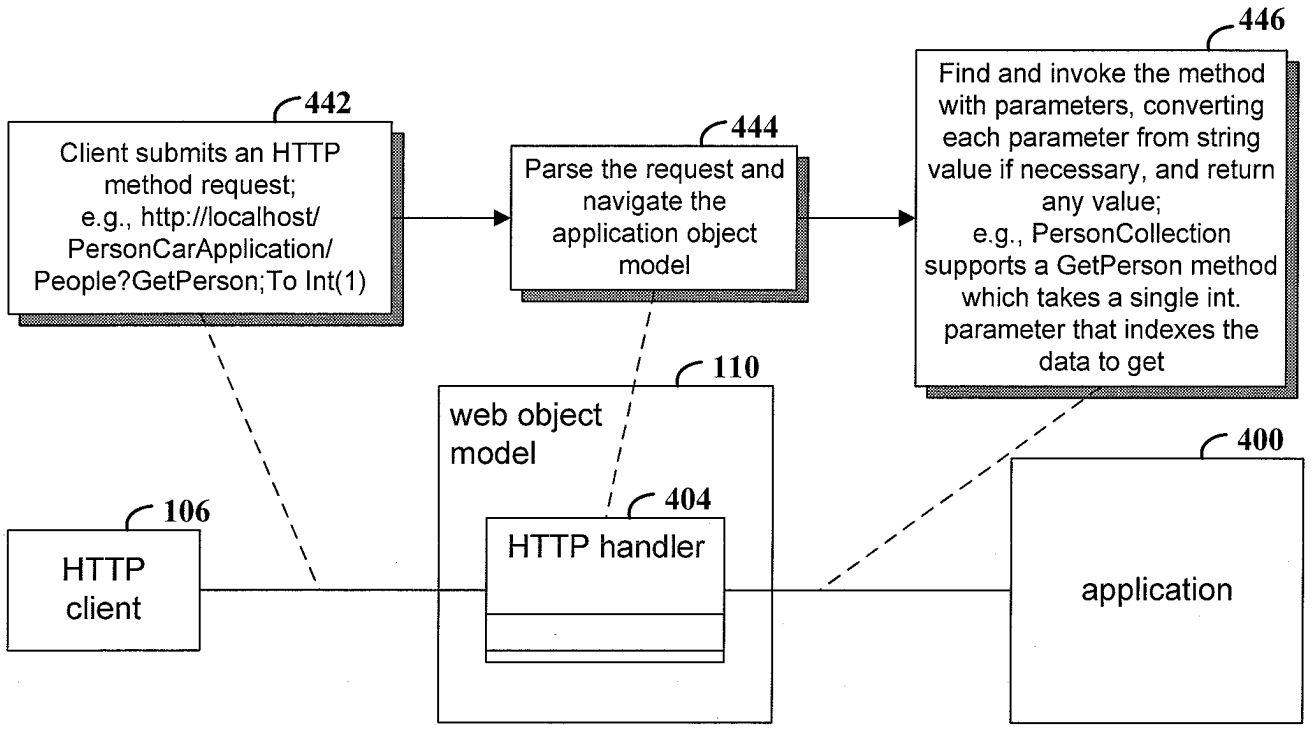
446
Find and invoke the method with parameters, converting each parameter from string value if necessary, and return any value; e.g., PersonCollection supports a GetPerson method which takes a single int. parameter that indexes the data to get
442
Client submits an HTTP method request; e.g., http://localhost/ PersonCarApplication/ People?GetPerson;To Int(1)
444
Parse the request and navigate the application object model
110
web object model
400
106
HTTP client
404
HTTP handler
application

FIG. 4D

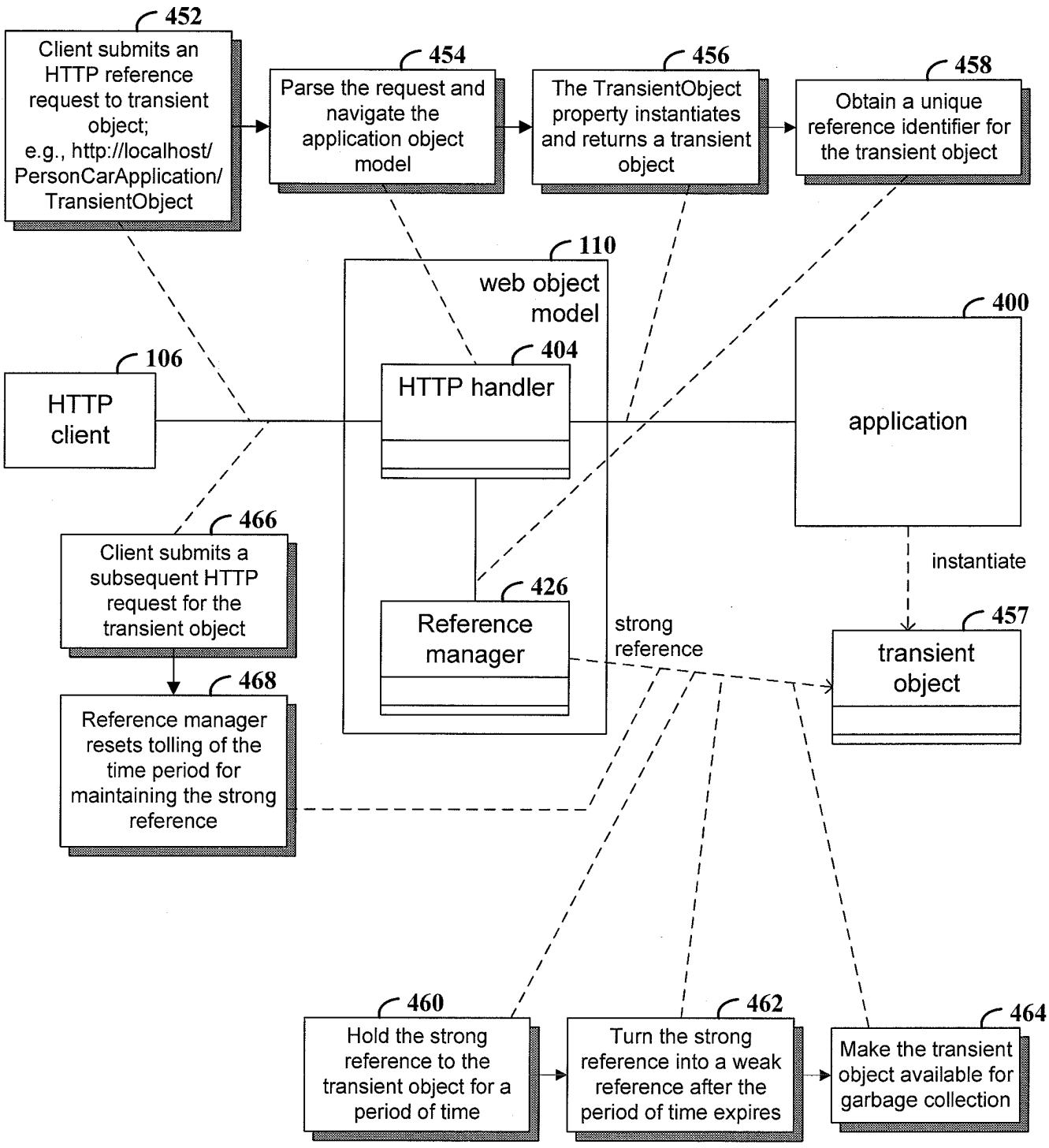
452
Client submits an HTTP reference request to transient object; e.g., http://localhost/PersonCarApplication/TransientObject
454
Parse the request and navigate the application object model
456
The TransientObject property instantiates and returns a transient object
458
Obtain a unique reference identifier for the transient object
110
web object model
400
106
HTTP client
404
HTTP handler
application
466
Client submits a subsequent HTTP request for the transient object
426
Reference manager
strong reference
instantiate
457
transient object
468
Reference manager resets tolling of the time period for maintaining the strong reference
460
Hold the strong reference to the transient object for a period of time
462
Turn the strong reference into a weak reference after the period of time expires
464
Make the transient object available for garbage collection

FIG. 4E

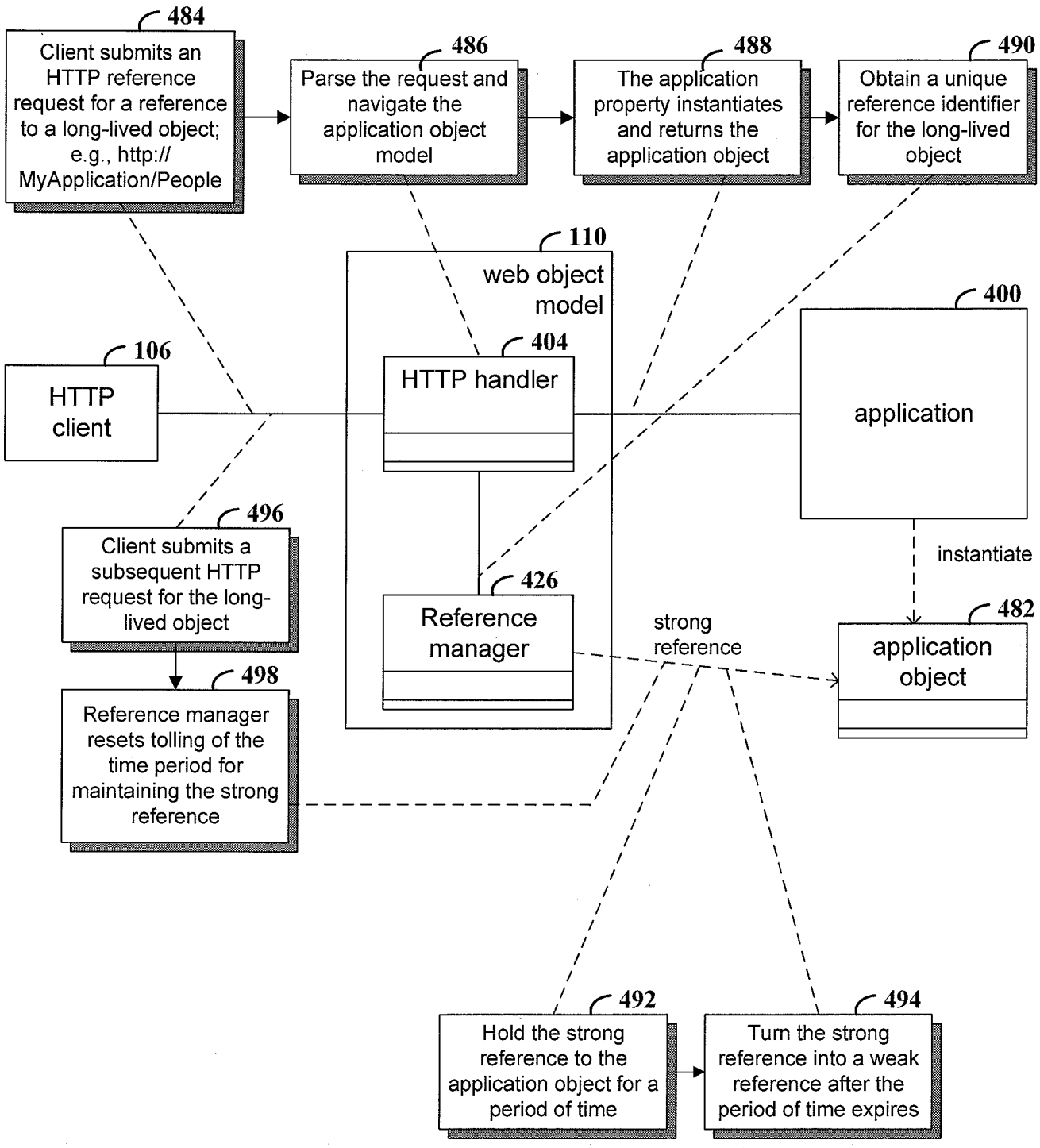
484
Client submits an HTTP reference request for a reference to a long-lived object; e.g., http:// MyApplication/People
486
Parse the request and navigate the application object model
488
The application property instantiates and returns the application object
490
Obtain a unique reference identifier for the long-lived object
110
web object model
400
106
HTTP client
404
HTTP handler
application
496
Client submits a subsequent HTTP request for the long-lived object
426
Reference manager
instantiate
482
application object
strong reference
498
Reference manager resets tolling of the time period for maintaining the strong reference
492
Hold the strong reference to the application object for a period of time
494
Turn the strong reference into a weak reference after the period of time expires

FIG. 4F

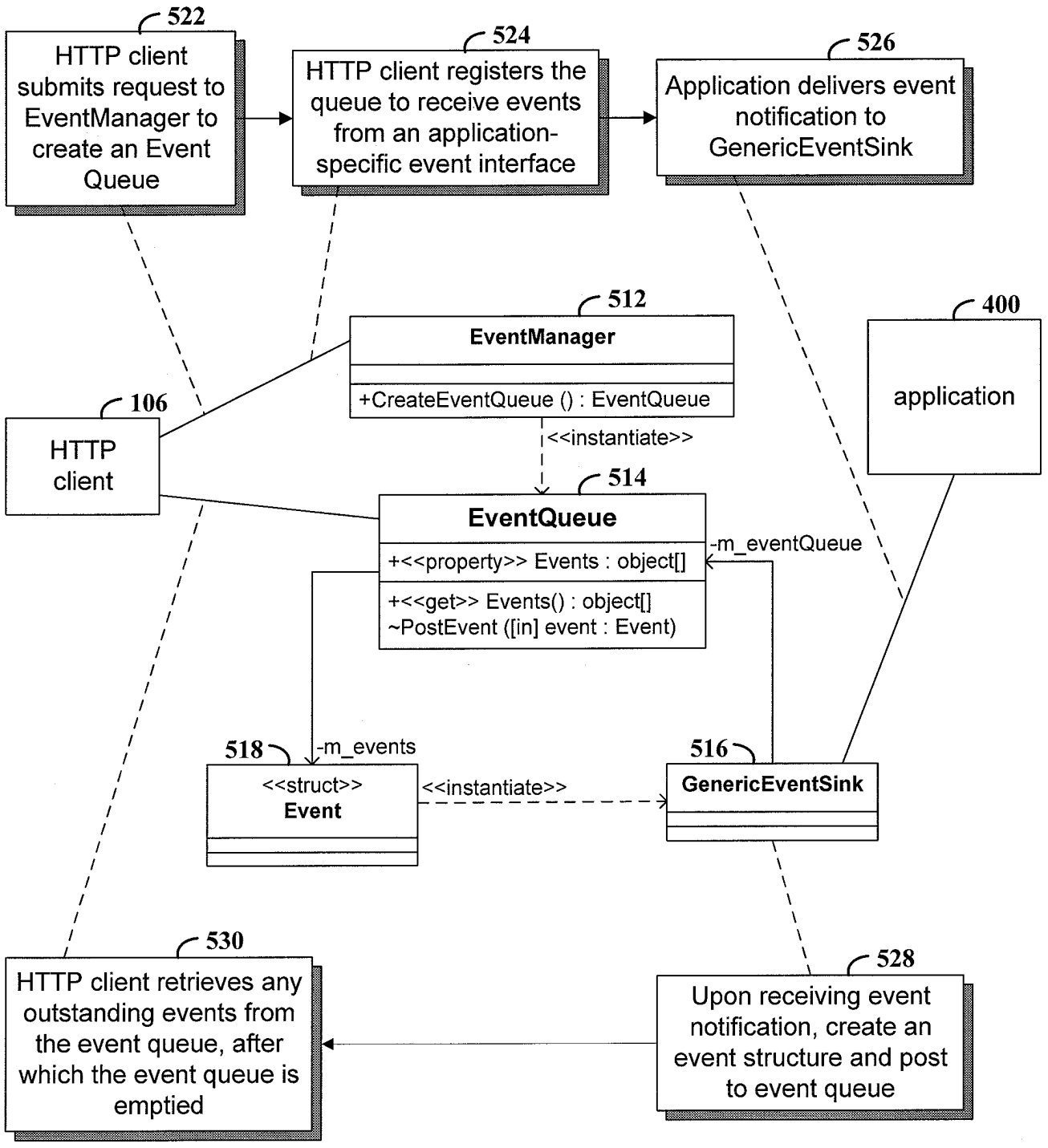
522
HTTP client submits request to EventManager to create an Event Queue
524
HTTP client registers the queue to receive events from an application-specific event interface
526
Application delivers event notification to GenericEventSink
512
EventManager
+CreateEventQueue () : EventQueue
400
application
106
HTTP client
<<instantiate>>
514
EventQueue
+<<property>> Events : object[]
+<<get>> Events() : object[]
~PostEvent ([in] event : Event)
-m_eventQueue
518
-m_events
<<struct>>
Event
<<instantiate>>
516
GenericEventSink
530
HTTP client retrieves any outstanding events from the event queue, after which the event queue is emptied
528
Upon receiving event notification, create an event structure and post to event queue

FIG. 4G